Early Stopping Byzantine Agreement in $(1 + \epsilon) \cdot f$ Rounds

Fatima Elsheimy[1], Julian Loss[2], and Charalampos Papamanthou[1]

[1] Yale University
[2] CISPA Helmholtz Center for Information Security

Abstract. In this paper, we present two *early stopping* Byzantine agreement protocols in the authenticated setting against a corrupt minority $t < n/2$, where t represents the maximum number of malicious parties. Early stopping protocols ensure termination within a number of rounds determined solely by the actual number of malicious nodes f present during execution, irrespective of t.
Our first protocol is deterministic and ensures early stopping termination in $(d + 5) \cdot (\lfloor f/d \rfloor + 3)$ rounds, where d is a fixed constant. For example, for all $d \geq 6$, our protocol runs in at most $(1 + \epsilon) \cdot f$ rounds (where $0 < \epsilon < 1$), improving (for large f) upon the best previous early stopping deterministic broadcast protocol by Perry and Toueg [1], which terminates in $min(2f + 4, 2t + 2)$ rounds. Additionally, our second protocol is randomized, ensuring termination in an expected constant number of rounds and achieving early stopping in $(d + 9) \cdot (\lfloor f/d \rfloor + 2)$ rounds in the worst case. This marks a significant improvement over a similar result by Goldreich and Petrank. [2], which *always* requires an expected constant number of rounds and $O(t)$ rounds in the worst case, i.e., does not have the early stopping property.

1 Introduction

Byzantine Agreement (BA) is a fundamental problem in distributed computing. In the BA problem, n parties start with some value in $\{0, 1\}$ and wish to jointly agree on one value while tolerating up to $t < n/2$ Byzantine parties (Agreement.) If all honest parties start with the same value, they must output that value (Validity.) The foundations of this field were established by the pioneering work of Lamport, Shostak, and Pease in the 1980s [3]. One of the main metrics of efficiency for BA protocols is their *round complexity*, i.e., the number of synchronous interactions required for the protocol to terminate. This is the focus of our paper.

A seminal result by Dolev and Strong [4][3] demonstrates that any BA protocol capable of tolerating $t < n/2$ malicious parties necessitates at least $t + 1$ rounds

[3] [4] presents the result for Byzantine Broadcast, a variant of Byzantine Agreement in which a designated sender sends an input value to other parties who must reach consensus on this value. The resilience for Byzantine Broadcast is $t < n$. There is a known reduction to Byzantine Agreement with optimal resilience of $t < n/2$.

in some runs. However, this bound is considered loose for protocol executions where the number of corruptions, f, is less than t. According to Dolev et al. [5], the round complexity lower bound in this scenario is $\min\{f+2, t+1\}$. Thus, a series of works studying *early stopping protocols* which terminate in time that depends only on the actual number of corruptions f. For the information-theoretic setting and $t < n/3$, this has culminated in the work of Abraham and Dolev [6] who gave the first early stopping protocol with polynomial communication and optimal round complexity of $\min\{f+2, t+1\}$. By comparison, the authenticated setting (where signatures can be used) with $t < n/2$ malicious corruptions is far less explored. To the best of our knowledge, the only early stopping protocol in this setting is due to Perry and Toueg [1] which has (sub-optimal) round complexity $\min\{2f+4, 2t+2\}$. This raises the following natural question: *Is there an early-stopping protocol for authenticated Byzantine agreement with $t < n/2$ corruptions which approaches the lower bound of $\min\{f+2, t+1\}$?* We answer this question affirmatively by showing the following results:

- We begin by proving a deterministic early-stopping Byzantine agreement protocol that terminates in $(d+5) \cdot (\lfloor f/d \rfloor + 3)$ rounds, where d is a fixed positive constant. In particular, for all $d \geq 6$ and

$$f > \frac{3d^2 + 11d}{d-5}$$

  our protocol always outperforms Perry and Toueg's protocol. In general, our protocol achieves a round complexity of

$$(1 + O(1/d)) \cdot f + O(d)\,,$$

  which simplifies to $(1+\epsilon) \cdot f$ whenever d behaves as a constant in f.
- We then show an early stopping randomized Byzantine agreement protocol with expected constant rounds, whose worst-case round complexity is $(d+9) \cdot (\lfloor f/d \rfloor + 2)$, where again, d is a predefined constant. Our protocol compares favorably with protocols obtained via the generic compiler of Goldreich and Petrank [2]. Like our work, their compiler gives an expected constant round protocol, but its worst-case round complexity is $O(t)$—therefore it does not yield an early stopping protocol.

At the heart of our construction, we devise a novel method of eliminating faulty parties that keep the protocol from terminating. Our construction relies on prior work of Fitzi and Nielsen [7] to improve the ratio of eliminated parties to protocol rounds. On average, our protocol eliminates 1 party every $1 + 5/d$ rounds, whereas the protocol of Perry and Toueg's protocol eliminates 1 party every 2 rounds. We now explain our techniques in more detail.

1.1 Technical Overview

We give now give a more detailed overview over our techniques.

Correct-Or-Detect Broadcast. We begin by recalling the Correct-Or-Detect Broadcast protocol of Fitzi and Nielsen which forms the basis of our construction. Their protocol, henceforth denoted $\Pi_{\mathsf{d\text{-}CoD}}$ [7], is parametrized by an arbitrary positive integer d and a designated sender P_s and runs in $d+4$ rounds. $\Pi_{\mathsf{d\text{-}CoD}}$ is based on the seminal broadcast protocol of Dolev and Strong, which itself runs in $t+1$ rounds and is secure against any number of $t < n$ corrupted parties. Rather than achieving full broadcast, parties in $\Pi_{\mathsf{d\text{-}CoD}}$ terminate the protocol in two possible modes C (correct) and D (detect). In case an honest party terminates in mode C, $\Pi_{\mathsf{d\text{-}CoD}}$ achieves the properties of broadcast, i.e., all parties agree on the sender's value. Moreover, if the sender P_s is honest, all honest parties always terminate in mode C. On the other hand, if some honest party terminates in mode D, $\Pi_{\mathsf{d\text{-}CoD}}$ may not achieve the properties of broadcast. Yet, in this case, the protocol ensures that all parties identify a common set of d corrupted parties. To this end, every party P_i among the set of honest parties $\mathcal{H}$ outputs a list $\mathcal{F}_i$ of parties it knows to be corrupted, where the protocol ensures that $|\bigcap_{P_i\in\mathcal{H}} \mathcal{F}_i| \geq d$. It is important to note that there is no agreement among parties on what mode the protocol terminates in (otherwise, $\Pi_{\mathsf{d\text{-}CoD}}$ would be a full-fledged broadcast protocol). We extend the construction of Fitzi and Nielsen for binary messages to messages of arbitrary length in the straight-forward way by broadcasting a message bit by bit and determining the termination mode as C iff all of the bit-wise sub-instances output C. Otherwise, we output D and take the union of identified malicious sets output in any of these instances.

Graded Consensus with Detection. We now explain our main technical building block, which we refer to as *graded consensus with corruption*. For simplicity, we focus here on our basic version of this primitive in which all parties input a binary value v_i along with their current list $\mathcal{F}_i$ of faulty parties. We additionally require that honest parties are never in each others list of identified corrupted parties.

The protocol outputs a value $y_i \in \{0,1\}$ along with a grade $g_i \in \{0,1\}$ and an updated list $\mathcal{F}_i^*$ of faulty parties. As with existing constructions of graded consensus in the literature, our protocol uses the grade g_i to indicate a party's confidence in its output y_i. *Graded consistency* says that on outputting grade $g_i = 1$, P_i knows that all parties agree on P_i's output y_i, but they might not know that they agree (as they have output grade 0). On the other hand, we ensure *graded validity*: if all honest parties input the same value v to the protocol, then all honest parties output $y_i = v$ and grade $g_i = 1$.

The distinguishing feature of our new construction is to ensure that if two honest parties P_i and P_j disagree on their respective outputs $y_i \neq y_j$, then they identify a common set of at least d corrupted parties and extend their faulty lists $\mathcal{F}_i^*$ accordingly. Importantly, we can ensure that the intersection $\bigcap_{P_i\in\mathcal{H}} \mathcal{F}_i^*$ contains at least d corrupt parties that are not contained in the common set of parties' faulty input lists $\bigcap_{P_i\in\mathcal{H}} \mathcal{F}_i$. Because the faulty lists of honest parties can never contain honest parties, this automatically implies that parties agree on their output (albeit possibly with grade 0) once there are fewer than d malicious

parties. This property will be crucially exploited in our overall construction of Byzantine Agreement.

From CoD-Broadcast to Graded Consensus with Detection. Our construction is remarkably simple and builds on the multivalued CoD-Broadcast described earlier. Once again, our construction follows the ideas of Fitzi and Nielsen. At the onset of an iteration, we have parties issue to each other *proofs of participation (PoP)*, meaning that every party P_i sends a signature to every party P_j where P_j is *not* in the list $\mathcal{F}_i$ input to the graded consensus protocol. Since honest parties are never part of each other's faulty lists, every honest party obtains a PoP in this way, allowing it to participate in the rest of the protocol. On the other hand, parties who are commonly identified as corrupt do not obtain such a proof and are banned from participating in the protocol. To enforce this measure, we have every party send its PoP along with its input v_i via $\Pi_{\mathsf{d\text{-}CoD}}$ to all parties. To determine the output, we let parties take a majority over all the instances that were received with a proper PoP attached. On the other hand, we let parties update their faulty lists with malicious parties identified in instances regardless of whether they had a proper PoP attached to them. To output $y_i = v$ with grade $g_i = 1$, a party P_i waits to observe $t+1$ instances terminate on value v in mode C (and with a PoP attached). On the other hand, for grade $g_i = 0$, P_i simply takes the majority bit over all instances with a correct PoP attached (regardless of what mode they terminate in). From the properties of $\Pi_{\mathsf{d\text{-}CoD}}$, it immediately follows that the usual consistency and validity properties of graded consensus. On the other hand, disagreement can only happen if at least one of the $\Pi_{\mathsf{d\text{-}CoD}}$ instances terminates in mode D. In this case, all parties can update their lists $\mathcal{F}_i^*$ with a common set of at least d newly identified malicious parties. Moreover, our protocol adds only 1 round (for PoPs) to the running time of $\Pi_{\mathsf{d\text{-}CoD}}$, thus coming out to a total running time of $d + 5$ rounds.

From GC with Detection to Deterministic Byzantine Agreement. We run the detecting graded consensus protocol described above in iterations. In each iteration k, parties update their input $v_i, \mathcal{F}_i$ to the output value y_i and faulty list $\mathcal{F}_i^*$ of iteration $k-1$. A party P_i terminates after observing the graded consensus protocol outputting grade $g_i = 1$ in some iteration k and running for one more subsequent iteration. By graded validity, this ensures that parties all parties observe the same condition by iteration $k + 1$ and can terminate by iteration $k + 2$ at the latest. The detection property of our graded consensus module ensures that in every iteration where parties do not terminate, they all add d common parties to their list of identified corrupted parties. If there are less than d malicious parties left, honest parties still output the same value. Thus, after at most $\lfloor f/d \rfloor$ iterations, all remaining parties must output the same value. By the above argument, this ensures that they terminate within at most three more iterations; one iteration to output the same value and two more from the above argument. Since each iteration takes $d+5$ rounds, our running time comes out to $(d + 5) \cdot (\lfloor f/d \rfloor + 3)$ many rounds.

Randomized Early Stopping Agreement. We conclude by explaining how to randomize the protocol sketched above. In this manner, we obtain an expected constant round protocol which also has early stopping complexity $(d+9)\cdot(\lfloor f/d \rfloor + 2)$. To this end, we add a few rounds on top of our detecting graded consensus protocol so as to obtain a stronger version of graded consensus with three possible grades $0, 1$, and 2. Here grade 2 indicates the highest confidence in a binary output y_i and indicates agreement for any party who observes it. On the other hand, grade 1 leaves open the possibility that another honest party has grade 0, in which case its corresponding output is the default value $\perp$. Our construction also extends the properties of the detecting properties of the $(0, 1)$ graded consensus protocol described above in the natural way and ensures that once no corrupted parties remain, parties always agree on their output.

Using this strengthened version of detecting graded consensus, we are able to run a standard construction of randomized byzantine agreement from graded consensus. As before, we iterate instances of graded consensus and input the output from the current iteration to the next iteration. However, parties update their input to the next iteration to a common random coin whenever it outputs $\perp$ with grade 0 in some iteration of the protocol. If the coin agrees for all parties with some constant probability p, this ensures that parties agree on what they input to any iteration with probability at least $p/2$. Thus, parties terminate the protocol in $O(2/p) = O(1)$ expected iterations of constant round length. The exact round complexity in expectation is $((2/p) + 2)(d + 9)$, where $d + 9$ are the number of rounds in an iteration. On the other hand, we can argue along the same lines as for the deterministic case that all parties terminate in the worst case after $\lfloor f/d \rfloor + 2$ iterations, i.e., there are less than d dishonest parties left to obstruct termination.

1.2 Related Work

Byzantine agreement has been extensively studied since the pioneering work of Shostak, Pease, and Lamport [3]. Dolev and Strong [4] established a critical result, showing that any broadcast protocol tolerating $t < n$ malicious parties requires at least $t + 1$ rounds. However, this bound was later refined by Dolev et al. [5], who demonstrated that when the number of corruptions, f, is much less than t, the lower bound is $\min(f + 2, t + 1)$. Since then, significant progress has been made in developing early stopping protocols.

The first such protocol in the information-theoretic setting with optimal resilience $t < n/3$ was introduced by Berman et al.[8], though it suffered from exponential communication complexity. Garay and Moses later addressed this issue, presenting a Byzantine agreement protocol with polynomial-sized messages but slightly suboptimal early stopping round complexity of $\min(f+5, t+1)$[9,10]. More recently, Abraham and Dolev [6] achieved a breakthrough by developing the first early stopping protocol with polynomial communication, optimal resilience, and optimal round complexity of $\min(f+2, t+1)$. While the information-theoretic setting has seen extensive research, there has been limited work in the

authenticated setting with optimal resilience $t < n/2$. To the best of our knowledge, Perry and Toueg [1] provide the only authenticated early stopping protocol with polynomial communication and a round complexity of $\min(2f + 4, 2t + 2)$.

As for randomized protocols, it has been established that they can achieve an expected constant number of rounds in both the information-theoretic setting [11] and the authenticated setting [12,13,14]. However, these protocols have a negligible probability of very long runs due to their failure probability. Goldreich et al. [2] presented a method to eliminate the failure probability, achieving an expected constant round complexity and worst-case round complexity of $O(t)$ for up to $t < n/2$ corruptions—therefore it does not yield an early stopping protocol. A follow-up work further improved this, achieving expected constant round complexity and optimal worst-case complexity of $t + 1$ rounds for a worse resilience of $t < n/8$ [15]. Achieving expected constant round complexity, $t + 1$ rounds worst case, and optimal resilience $t < n/3$ remains unresolved. Importantly, this question remains open even without considering the early stopping worst-case round complexity. We note that it is possible to terminate randomized protocols in round complexity that is independent of the number of corrupted parties. However, in this case, the number of rounds always depends on the desired error probability δ of the protocol. This makes such protocols difficult to compare to early stopping protocols. In particular, early stopping protocols may require much fewer rounds to terminate when the number f of corruptions is low.

Other works [16,17,18] have explored early stopping protocols but in much weaker adversary settings, such as omission and crash adversary models. A recent of work of Loss and Nielsen [19] gives the first early stopping protocol for the dishonest majority setting with $t < n$ corruptions, albeit with significantly worse round complexity $O(\mathsf{min}\{f^2, t\})$.

1.3 Paper Organization

Section 2 provides definitions for Byzantine Agreement, $(0, 1)$, and $(0, 1, 2)$-Graded d-Detecting Byzantine Agreement as well as for the cryptographic primitives we use such as Signature schemes and common coin. In Section 3, we discuss the intuition and the construction of the deterministic early-stopping protocol, along with its correctness proof. In section 4, as well as the intuition and construction of the randomized protocol. We defer some supplementary protocols and definitions to the Appendix.

2 Preliminaries

We begin by introducing the model as well as basic definitions.

Network and Setup Assumptions. We assume a a fully connected network of pairwise, authenticated channels between n parties $\{P_1, ..., P_n\} = \mathcal{P}$. We consider the *synchronous network model* where all parties have access to a synchronized clock and there is a known upper bound Δ on the message delays of honest

parties. This allows parties to run protocols in a round-by-round fashion where rounds are of length Δ and any message that is sent by an honest party at the beginning of a round are delivered by the end of that round to all honest parties. Parties are assumed to have established a public key infrastructure (PKI) of a digital signature scheme that provides an efficient signing routine Sign and an efficient verification routine Verify. Every party P_i is associated with a public key pk_i that is known to all parties and where (only) P_i knows the corresponding secret key sk_i. This allows a party P_i to create a signature $\langle m \rangle_i$ on message m using its secret key sk_i via $\langle m \rangle_i := \mathsf{Sign}(\mathsf{sk}_i, m)$. $\langle m \rangle_i$ can then be efficiently verified by running $\mathsf{Verify}(\mathsf{pk}_i, \langle m \rangle_i, m)$. We refer to a signature $\langle m \rangle_i$ as *valid* if $\mathsf{Verify}(\mathsf{pk}_i, \langle m \rangle_i, m) = 1$. For ease of notation, we use the abbreviated notation $\langle m \rangle_i$ to refer to tuples $(m, \mathsf{sign}(m, \mathsf{sk}_i))$ throughout the paper.

Adversary Model. We consider an adaptive Byzantine adversary that can corrupt up to $t < n/2$ parties at any point of a protocol execution. We refer to the *actual number* of corruptions during an execution of the protocol as $f \leq t$. A corrupt (or malicious) party P_i is under full control of the adversary and may deviate arbitrarily from the protocol. In particular, the adversary learns P_i's signing key sk_i, which allows it to sign messages on P_i's behalf. In addition, we allow the adversary to delete (or replace with its own) any undelivered messages of a newly corrupted party P_i that P_i sent while it was still honest. We denote the set of uncorrupted (or honest) parties as $\mathcal{H}$.

We assume that the adversary is computationally bounded and cannot forge signatures of honest parties. In line with the literature in this area, we treat signatures as idealized primitives with perfect security. When instantiating the signature scheme with an existentially unforgeable one, we obtain protocols with non-neglible probability of failure.

Common Coin. We assume an ideal coin-flip protocol $\mathsf{CoinFlip}$ that allows parties to agree with constant probability $p < 1$ on a random coin in $\{0, 1\}$. This protocol can be viewed as an ideal functionality [20] that upon receiving input r from $t + 1$ parties generates a random coin c_i and sends $(c_i^{(r)})$ to each party $P_i \in \mathcal{P}$, where $c_i^{(r)} = c_j^{(r)}$ with probability at least p. The value remains uniform from the adversary's view until the first honest party has queried $\mathsf{CoinFlip}$. Such a primitive can be achieved using verfiable random functions [21], threshold signatures [22], or verifiable secret sharing [14].

We begin by presenting definitions of well-known primitives, such as Byzantine agreement and graded consensus. Following this, we introduce new definitions for our proposed protocols: graded consensus with detection.

Definition 1 (Byzantine Agreement). *Let Π be protocol executed among parties $P_1, ..., P_n$, where each party P_i holds an input $v_i \in \{0, 1\}$ and outputs a value $y_i \in \{0, 1\}$ upon terminating. A protocol Π achieves* Byzantine Agreement*, if the following properties hold whenever at most t parties are corrupted.*

- *Validity: If every honest party P_i inputs $\mathsf{v}_i = \mathsf{v}$, then all honest parties output $y_i = \mathsf{v}$;*

- *Consistency: All honest parties output the same value v.*
- *Termination: Every honest party terminates.*

Definition 2 ($(0,1,2)$-Graded Agreement). *Let Π be a protocol executed by parties $P_1, ..., P_n$, where each party P_i inputs $v_i \in \{0,1,2\}$ and outputs a value $y_i \in \{0,1,\perp\}$ and a grade $g_i \in \{0,1\}$ upon terminating. A protocol Π achieves* $(0,1,2)$-Graded Agreement *if the following properties hold whenever at most t parties are corrupted.*

- *Graded Validity: If all honest parties P_i have the same input value $v_i = \mathsf{v}$ then all honest parties output $y_i = \mathsf{v}$ and $g_i = 2$*
- *Graded Consistency: Let P_i and P_j denote honest parties that output y_i, g_i and y_j, g_j, respectively. Then (1) $|g_i - g_j| \leq 1$ and (2) $g_i, g_j \geq 1$ implies that $v_i = v_j$*
- *Termination: Every honest party terminates.*

Definition 3 (Correct or Detect Broadcast (d-CoD)). *Let Π be protocol executed by parties $P_1, ..., P_n$ where a designated sender P_s holds input $v \in \{0,1\}^*$ and each party P_i outputs a value $y_i \in \{0,1\}^*$, a list of faulty parties $\mathcal{F}_i \subseteq \mathcal{P}$, and a flag $det_i \in \{C, D\}$ upon terminating. Π achieves* Correct or Detect Broadcast (CoD)*, if the following properties hold whenever at most t parties are corrupted.*

- *$\mathcal{F}$-soundness: If an honest party P_i outputs $\mathcal{F}_i$, then $\mathcal{F}_i$ consists only of corrupted parties.*
- *Consistency: If $det_i = C$ for some honest party P_i, then every honest party P_j outputs $y_j = y_i$. In this case, we say that the protocol has* correctness.
- *Validity: If P_s is honest and inputs v, then every honest party P_i outputs $(y_i = v, \mathcal{F}_i = \emptyset, det_i = C)$.*
- *d-Detection: If for some honest party P_i, $det_i = D$, then $|\bigcap_{P_j \in H} \mathcal{F}_j| \geq d$. In this case, we say that the protocol has* detection.
- *Termination: Every honest party terminates.*

Definition 4 ($(0,1)$-Graded d-Detecting Agreement). *Let Π be a protocol executed by parties $P_1, ..., P_n$, where each party P_i inputs $v_i \in \{0,1\}$ and a list of faulty parties $\mathcal{F}_i \subset \mathcal{P}$ and outputs a value $y_i \in \{0,1\}$, a grade $g_i \in \{0,1\}$, and an updated faulty list $\mathcal{F}_i^\star \subset \mathcal{P}$ upon terminating. Π achieves* $(0,1)$-Graded d-Detecting Agreement *if the following properties hold whenever at most t parties are corrupted and for all honest parties P_i, $\mathcal{F}_i$ contains only corrupted parties.*

- *Graded Validity: If all honest parties P_i have the same input value $v_i = \mathsf{v}$ then all honest parties output $y_i = \mathsf{v}$ and $g_i = 1$*
- *Graded Consistency: If two honest parties P_i and P_j output $g_i = g_j = 1$, respectively, then $y_i = y_j$*
- *d-Detection: If two honest parties P_i and P_j output $y_i = 1$ and $y_j = 0$, respectively, then an additional d parties are added to the faulty lists of all honest parties; that is, $\left|\left(\bigcap_{P_j \in \mathcal{H}} \mathcal{F}_j^\star\right) \setminus \left(\bigcap_{P_j \in \mathcal{H}} \mathcal{F}_j\right)\right| \geq d$.*
- *Soundness: If an honest party P_i outputs $\mathcal{F}_i^\star$, then $\mathcal{F}_i^\star$ consists only of corrupted parties. Furthermore, $\mathcal{F}_i \subseteq \mathcal{F}_i^\star$*

– *Termination: Every honest party terminates.*

Definition 5 ($(0,1,2)$-Graded d-Detecting Agreement). *Let Π be a protocol executed by parties $P_1, ..., P_n$, where each party P_i inputs $v_i \in \{0,1,2\}$ and a list of faulty parties $\mathcal{F}_i \subset \mathcal{P}$ and outputs a value $y_i \in \{0,1,\perp\}$, a grade $g_i \in \{0,1\}$, and an updated faulty list $\mathcal{F}_i^\star \subset \mathcal{P}$ upon terminating. A protocol Π achieves* $(0,1,2)$-Graded d-Detecting Agreement *if the following properties hold whenever at most t parties are corrupted and for all honest parties P_i, $\mathcal{F}_i$ contains only corrupted parties.*

– *Graded Validity: If all honest parties P_i have the same input value $v_i = v$ then all honest parties output $y_i = v$ and $g_i = 2$*
– *Graded Consistency: Let P_i and P_j denote honest parties that output y_i, g_i and y_j, g_j, respectively. Then (1) $|g_i - g_j| \leq 1$ and (2) $g_i, g_j \geq 1$ implies that $v_i = v_j$*
– *d-Detection: If any honest party P_i outputs $g_i < 2$, then an additional d parties are added to the faulty lists of all honest parties; that is, $\left|\left(\bigcap_{P_j \in \mathcal{H}} \mathcal{F}_j^\star\right) \setminus \left(\bigcap_{P_j \in \mathcal{H}} \mathcal{F}_j\right)\right| \geq d$.*
– *Soundness: If an honest party P_i outputs $\mathcal{F}_i^\star$, then $\mathcal{F}_i^\star$ consists only of corrupted parties. Furthermore, $\mathcal{F}_i \subseteq \mathcal{F}_i^\star$.*
– *Termination: Every honest party terminates.*

Definition 6 (Signature Chain). *Let $m \in \{0,1\}^*$, let $k \in \mathbb{N}$, and let σ denote a sequence k distinct numbers $j_1, ..., j_k \in [n]$. We write $\langle m \rangle_\sigma$ to denote the nested signatures $\langle \ldots \langle m \rangle_{j_1} \ldots \rangle_{j_k}$ and refer to σ as a* signature chain of length k. *$\langle m \rangle_\sigma$ is said to be* valid *if for all k, the signature with respect to pk_{j_k} is valid.*

3 Deterministic Early-Stopping Byzantine Agreement

As previously discussed, both of our early-stopping protocols are constructed based on the $(0,1)$-Graded d-Detecting Byzantine Agreement protocol, which in turn is derived from Correct-or-Detect Broadcast protocol $\Pi_{\mathsf{d\text{-}CoD}}$ [7] and Proof of Participation protocol Π_{PoP} [7]. We adopt a bottom-up approach, initially introducing the aforementioned subroutines and subsequently demonstrating the construction of $(0,1)$-Graded d-Detecting Agreement protocol and our early-stopping protocols.

3.1 Correct or Detect Broadcast Protocols ($\Pi_{\mathsf{d\text{-}CoD}}$ and $\Pi_{\mathsf{d\text{-}MCoD}}$)

In essence, $\Pi_{\mathsf{d\text{-}CoD}}$ (Fig. 1) is a broadcast protocol that ensures that all parties either agree on the sender's value or, alternatively, all honest parties identify a common set of d corrupted parties. The parameter d is a predefined constant that directly impacts the protocol's round complexity, with the protocol running for $d+4$ rounds. $\Pi_{\mathsf{d\text{-}CoD}}$ is 1-biased, meaning the designated sender P_s sends his value in the first round only if it is $v_s = 1$; otherwise, he refrains from sending anything if it is $v_s = 0$. Essentially, $\Pi_{\mathsf{d\text{-}CoD}}$ is a modified version of Dolev-Strong [4] that's

1- biased and forced to terminate in $d+4$ rounds. In every round $r > 1$, if a party P_i receives a message $\langle 1 \rangle_\sigma$ with $r = |\sigma|$ valid signatures, including the sender's signature for the first time, it accepts the message, appends its own signature, and forwards it to all parties in the next round. Let r_i be the first round where party p_i receives such a message. P_i sets $y_i \in \{0, 1\}$ and $det_i \in \{C, D\}$ based on the value of r_i. If $r_i \leq d + 1$ or $d + 4$, P_i outputs $det_{=}C$; otherwise, it outputs $det_{=}D$. If $r_i \leq d + 2$, it outputs $y_i = 1$; otherwise, it outputs $y_i = 0$. For completeness, we show the $\Pi_{\mathsf{d\text{-}CoD}}$ protocol in fig. 1 and state the correctness lemma (Lemma 1) for $\Pi_{\mathsf{d\text{-}CoD}}$. We refer the reader to [7] for the full proof.

Protocol $\Pi_{\mathsf{d\text{-}CoD}}$

- **Input and Initialization:** If $P_i = P_s$, let v_s denote P_s's input. P_i sets $\mathcal{F}_i := \emptyset$ $y_i := 0$, $det_i := C, r_i := d + 4$.
- **Round** 1 **($P_i = P_s$):** If the sender's initial value is $v_s = 1$, it sends $\langle 1 \rangle_s$ to all parties. (Otherwise, it does nothing.)
- **Rounds** $r = 2$ **to** $d + 4$ **($P_i \neq P_s$):**
  - If P_i received a valid signature chain $\langle 1 \rangle_\sigma$ of length $r - 1$ in the previous round and $r_i = r - 2$, it appends to the chain its signature, i.e., it computes $\sigma' := \langle \langle 1 \rangle_\sigma \rangle_i$ and sends σ' to all parties.
  - If P_i receives a valid signature chain $\langle 1 \rangle_\sigma$ of length r and $r_i = d+4$, it sets $r_i := r - 1$. Furthermore, for $\sigma = s, \ldots, k, j$, P_i adds every party in $P_s, \ldots, P_k$ to $\mathcal{F}_i$.
- **Output Determination:** If $P_i = P_s$, P_i sets $y_i := v_s, det_i := C$ and terminates. Else if $r_i \leq d + 2$, party P_i sets $y_i := 1$. Else if $d + 2 \leq r_i \leq d + 3$, it sets $det_i := D$. Finally, P_i outputs $y_i, det_i, \mathcal{F}_i$ and terminates.

Fig. 1. Code of $\Pi_{\mathsf{d\text{-}CoD}}$ for party P_i.

Lemma 1. *$\Pi_{\mathsf{d\text{-}CoD}}$ achieves d-CoD as per Definition. 3 in $d + 4$ rounds.*

Next, we construct a protocol, $\Pi_{\mathsf{d\text{-}MCoD}}$ (see Fig. 2), that extends the binary input range of $\Pi_{\mathsf{d\text{-}CoD}}$ to a multivalued range. To achieve this, multiple $\Pi_{\mathsf{d\text{-}CoD}}$ protocols can be executed concurrently, allowing the sender to send each bit of their message string. Due to the concurrent execution, the resultant protocol still runs in $d + 4$ rounds; however, the communication complexity increases proportionally with the input size.

For a party p_i to output $det_i = C$, all concurrently invoked $\Pi_{\mathsf{d\text{-}CoD}}$ instances must terminate with $det_i = C$. Otherwise, the party outputs $det_i = D$. The output value y_i is obtained by concatenating all output bits from each $\Pi_{\mathsf{d\text{-}CoD}}$ instance. The output faulty list $\mathcal{F}_i$ is the union of all faulty lists produced by each invoked instance of $\Pi_{\mathsf{d\text{-}CoD}}$.

In the following lemma, we prove the correctness of $\Pi_{\mathsf{d\text{-}MCoD}}$ per Definition 3

Lemma 2. *$\Pi_{\mathsf{d\text{-}MCoD}}$ achieves d-CoD as per Definition 3 and terminates in $d + 4$ rounds.*

Protocol $\Pi_{\mathsf{d\text{-}MCoD}}$

- **Input and Initialization:** If $P_i = P_s$, let v_s denote P_s's input and set $l := |v_s|$. Set $\mathcal{F}_i := \emptyset, y_i := \perp, det_i := D, r_i := d+4$
- **Rounds** $r = 1$ **to** $d+4$**:**
  - Party P_s invokes in parallel l instances of $\Pi_{\mathsf{d\text{-}CoD}}$, where the input for the jth instance is bit $v_i[j], j \in [l]$. Let $(y^j_{CoD}, det^j_i, \mathcal{F}^j_i)$ denote the output of the jth instance for party P_i
- **Output Determination:** If $det^j_i = C$ for all $j \in [l]$, P_i sets $det_i := C$. It sets $y_i := y^1_{CoD} \parallel \cdots \parallel y^l_{CoD}$ and$\mathcal{F}_i = \bigcup_{j=1}^{l} \mathcal{F}^j_i$. P_i outputs $(y_i, det_i, \mathcal{F}_i)$ and terminates.

Fig. 2. Code of $\Pi_{\mathsf{d\text{-}MCoD}}$ for party P_i.

Proof. **$\mathcal{F}$-soundness:** The output faulty list $\mathcal{F}_i$ is the union of all faulty lists $\mathcal{F}^j_i$ produced by the l parallel invocations of $\Pi_{\mathsf{d\text{-}CoD}}$. Based on the $\mathcal{F}$-soundness of the $\Pi_{\mathsf{d\text{-}CoD}}$ protocol, the resulting faulty list $\mathcal{F}_i$ contains only malicious parties.
Consistency: If an honest party P_i outputs $det_i = C$, then for each $j \in [l]$, $det^j_i = C$. Thus, by consistency of $\Pi_{\mathsf{d\text{-}CoD}}$, each party P_j outputs the same bits in each of the l parallel instances of $\Pi_{\mathsf{d\text{-}CoD}}$ as party P_i. Since the output y_i is the concatenation of all output bits $y^b_{CoD} b = 1^l$, party P_j will output $y_j = y_i$.
Validity: If P_s is honest, it follows the same logic as discussed earlier since the output value y_i is simply the concatenation of the output values of all invoked $\Pi_{\mathsf{d\text{-}CoD}}$ instances and $det_i = C$ holds if for each instance j among those instances, $det^j_i = C$. Thus, validity follows directly from validity of $\Pi_{\mathsf{d\text{-}CoD}}$.
d-Detection: For a party to output $det_i = D$, at least one instance $j \in [l]$ among the l parallel instances of $\Pi_{\mathsf{d\text{-}CoD}}$ output $det^j_i = D$. Thus, the d-Detection property of $\Pi_{\mathsf{d\text{-}CoD}}$ implies that at least d malicious parties are added to every honest party P_i's faulty list $\mathcal{F}_i$ via $\mathcal{F}^j_i$.
Termination: $\Pi_{\mathsf{d\text{-}MCoD}}$ consists of concurrent instances of $\Pi_{\mathsf{d\text{-}CoD}}$. Based on the assumption that $\Pi_{\mathsf{d\text{-}CoD}}$ terminates, $\Pi_{\mathsf{d\text{-}MCoD}}$ will also terminate.
Round Complexity. $\Pi_{\mathsf{d\text{-}MCoD}}$ consists of concurrent execution of $\Pi_{\mathsf{d\text{-}CoD}}$, which runs in $d+4$ rounds. □

3.2 Proof of Participation (Π_{PoP})

At a high level, the Proof of Participation protocol, Π_{PoP}, allows each party to obtain a proof of its honesty, (PoP). A proof of participation, PoP, is considered *valid* if it is comprised of $t+1$ valid signatures from distinct parties $P_j \in \mathcal{P}$ of the form $\langle P_i \rangle_j$. To generate such a proof, each party P_i executes Π_{PoP} on input $\mathcal{F}_i$, where $\mathcal{F}_i$ represents its view of faulty parties. In the first round of Π_{PoP}, each party sends a message to all parties not in $\mathcal{F}_i$, asserting their honesty. If a party P_j receives at least $t+1$ such messages, it uses them as its proof.

Note that Π_{PoP} is associated with a session identifier ssid, which indicates the session in which it is invoked. This protocol generates a proof, PoP, based on the messages exchanged during that session. As a result, a proof PoP valid

for ssid is not applicable to any other session identifier $\mathsf{ssid}' \neq \mathsf{ssid}$. Lastly, we define two primary properties of Π^k_{PoP} in Lemma 3 and Lemma 4.

Protocol Π_{PoP}

- **Input and Initialization:** Let $\mathcal{F}_i$ denote P_i's input. P_i sets $\mathsf{PoP}_i := \perp$
- **Round 1:**
  - For each party $P_j \notin \mathcal{F}_i$, party P_i sends $\langle P_j \rangle_i$ to party P_j
- **Output Determination:** If P_i receives valid signatures $\langle P_i \rangle_j$ from at least $t+1$ distinct parties, P_i collects these messages into PoP_i. P_i outputs PoP_i and terminates.

Fig. 3. Code of Π_{PoP} for party P_i.

Lemma 3. *Assume no honest party P_j is in the faulty list $\mathcal{F}_i$ of any other honest party P_i. Then, each honest party P_j outputs a valid PoP_j.*

Proof. There are at most $t < n/2$ malicious parties. Each honest party P_i sends $\langle P_j \rangle_i$ to every party $P_j \notin \mathcal{F}_i$. As per assumption, every honest party p_i will receive at least $t+1$ messages of $\langle p_i \rangle_j$. Consequently, every honest party sets its output PoP_i to the aggregation of those received messages. □

Lemma 4. *Assume there exists some party P_j such that $P_j \in \mathcal{F}_i$ for all honest parties $P_i \in \mathcal{P}$. Then, P_j does not output a valid PoP_j.*

Proof. There are at most $t < n/2$ malicious parties. No honest party will send $\langle p_j \rangle_i$ to $P_j \in \mathcal{F}_i$. Thus, P_j can collect at most $t < n/2$ such messages, which are not enough to form PoP_j. □

3.3 $(0,1)$-Graded d-Detecting Agreement Construction ($\Pi_{\mathsf{1\text{-}GDA}}$)

In summary, $\Pi_{\mathsf{1\text{-}GDA}}$ (see Fig.4) is a variant of Graded Consensus protocols[11]. However, in $\Pi_{\mathsf{1\text{-}GDA}}$, honest parties also output a list of detected malicious parties. $\Pi_{\mathsf{1\text{-}GDA}}$ ensures that either every honest party outputs the same value y_i, or every honest party identifies at least d malicious parties (achieving d-detection).

In $\Pi_{\mathsf{1\text{-}GDA}}$, each party starts on $v_i \in \{0,1\}$ and faulty list $\mathcal{F}_i$. Each party outputs a value $y_i \in \{0,1\}$, a grade $g_i \in \{0,1\}$, and an updated list of identified malicious parties $\mathcal{F}_i^\star \subset \mathcal{P}$. In the first round, each party P_i executes Π_{PoP} to obtain a valid PoP_i. Consequently, each party P_i invokes $\Pi_{\mathsf{d\text{-}MCoD}}$ with input (PoP_i, v_i). For simplicity, we denote in the following $\Pi^j_{\mathsf{d\text{-}MCoD}}$ as the protocol instance in which P_j is the sender. Each party stores the output $((\mathsf{PoP}_{i,j}, y_{i,j}), det_i^j, \mathcal{F}_i^j)$ from all terminated instances of $\Pi^j_{\mathsf{d\text{-}MCoD}}$ for each $P_j \in \mathcal{P}$. Consequently, party P_i maintains a list H_i of all parties P_j that sent a valid PoP_j via $\Pi^j_{\mathsf{d\text{-}MCoD}}$. Each party P_i takes the union of all the faulty lists output by all $\Pi_{\mathsf{d\text{-}MCoD}}$ instances to form $\mathcal{F}_i^\star$ in addition to the parties in its initial faulty list $\mathcal{F}_i$.

To determine the output value y_i and grade g_i, a party P_i only considers the output of $\Pi^j_{\mathsf{d\text{-}MCoD}}$ from parties P_j in H_i. If there is a bit $\mathsf{v} \in \{0,1\}$ such that for at least $t+1$ of the parties $P_j \in H_i$, $y^j_{mCoD} = \mathsf{v}$ and $det^j_i = C$, party P_i sets its output to $y_i = \mathsf{v}$ and $g_i = 1$. Otherwise, if no such $t+1$ parties exist, P_i outputs the majority value over values $y_{i,j}$ among parties P_j in H_i. The protocol runs for $d+5$ rounds: one round for Π_{PoP} and $d+4$ for $\Pi_{\mathsf{d\text{-}MCoD}}$.

Protocol $\Pi_{\mathsf{1\text{-}GDA}}$

- **Input and Initialization:** Let v_i and $\mathcal{F}_i$ denote P_i's input. P_i sets $y_i := v_i, g_i := 0, H_i, \mathcal{F}^\star_i := \emptyset$, and $y_{i,j}, \mathsf{PoP}_{i,j} := \perp$ for $j \in [n]$
- **Round 1:**
  - Party P_i runs Π_{PoP} on input $\mathcal{F}_i$. Let PoP_i denote the output.
- **Rounds $r = 2$ to $d+5$:**
  - Party P_i invokes $\Pi_{\mathsf{d\text{-}MCoD}}$ on input (PoP_i, v_i). Denote the instance of $\Pi_{\mathsf{d\text{-}MCoD}}$ in which P_j is the sender as $\Pi^j_{\mathsf{d\text{-}MCoD}}$.
  - Party P_i stores the output of $\Pi^j_{\mathsf{d\text{-}MCoD}}$ for $P_j \in \mathcal{P}$; $((\mathsf{PoP}_{i,j}, y_{i,j}), det^j_i, \mathcal{F}^j_i) := \Pi^j_{\mathsf{d\text{-}MCoD}}$
  - For each party P_j such that $\mathsf{PoP}_{i,j}$ is a *valid* PoP, P_i adds P_j to H_i.
- **Output Determination:**
  - Party P_i accumulates the faulty lists of all instances of $\Pi^j_{\mathsf{d\text{-}MCoD}}$ along with the input $\mathcal{F}_i$ as $\mathcal{F}^\star_i = \bigcup_{j\in[n]} \mathcal{F}^j_i \cup \mathcal{F}_i$.
  - If there exists $\mathsf{v} \in \{0,1\}$ and at least $t+1$ instances of $\Pi^j_{\mathsf{d\text{-}MCoD}}$ for $P_j \in H_i$ that terminate with $det^j_i = C$ and output $y_{i,j} = \mathsf{v}$, then P_i sets the output value $y_i := \mathsf{v}$ and the grade $g_i := 1$.
  - Otherwise, P_i sets $y_i := \mathsf{v}$, where v is the the majority bit among values $y_{i,j}$ where $P_j \in H_i$.
  - Party P_i outputs $y_i, g_i, \mathcal{F}^\star_i$ and terminates.

Fig. 4. Code of $\Pi_{\mathsf{1\text{-}GDA}}$ for party P_i.

We start with proving graded validity.

Lemma 5. *$\Pi_{\mathsf{1\text{-}GDA}}$ achieves graded validity as per Definition 4.*

Proof. Assume that for all honest parties P_i, $v_i = \mathsf{v}$. Further, assume that for each honest party P_i, $P_i \notin \mathcal{F}_j$ for any honest party P_j. In the first round, every honest party P_i invokes Π_{PoP} on input $\mathcal{F}_i$. From assumption, every honest party P_i outputs a valid PoP_i according to lemma 3. Consequently, each party P_i invokes as the sender, $\Pi_{\mathsf{d\text{-}MCoD}}$ on input $(\mathsf{PoP}_i, \mathsf{v})$. According to the graded validity of $\Pi_{\mathsf{d\text{-}MCoD}}$ (Definition 3), if P_i is honest, each honest party P_j outputs $det^i_j = C$ and $y_{j,i} = v$. Furthermore, by assumption, P_i sends a valid PoP_i and thus, every honest P_i will add P_i to the list H_j. Thus, since there are at most $t < n/2$ malicious parties, every honest party P_i will output $(\mathsf{v}, C, \mathcal{F}^j_i)$ from at least $t+1$ instances of $\Pi_{\mathsf{d\text{-}MCoD}}$ for parties $P_j \in H_i$. Consequently, each honest

party sets $y_i = \mathsf{v}$ and $g_i = 1$.

In Lemma 6, we prove graded consistency.

Lemma 6. *$\Pi_{\text{1-GDA}}$ achieves graded consistency as per Definition 4.*

Proof. Assume that for each honest party P_i, $P_i \notin \mathcal{F}_j$ for any honest party P_j. A party P_i outputs $y_i = \mathsf{v}$ and $g_i = 1$ if at least $t+1$ instances $\Pi^j_{\text{d-MCoD}}$ corresponding to parties $P_j \in H_i$ terminate with $det^j_i = C$, and have the same output value $y_{i,j} = \mathsf{v}$. From consistency of $\Pi_{\text{d-MCoD}}$, every other honest party P_j outputs $y_{j,k} = \mathsf{v}$ for the same instances and adds the corresponding parties to those instances to H_j as they receive a *valid* PoP. Since $t < n/2$, the majority bit over all values $y_{j,k}, k \in H_j$ is also equal to v for every honest party P_j. Consequently, each honest party sets $y_i = \mathsf{v}$.

We proceed to prove the d-detection property.

Lemma 7. *$\Pi_{\text{1-GDA}}$ achieves d-detection as per Definition 4.*

Proof. Assume that for each honest party P_i, $P_i \notin \mathcal{F}_j$ for any honest party P_j. Note, for any party P_k that is in $\mathcal{F}_i$ of all honest parties $P_i \in \mathcal{P}$, P_k can not obtain a *valid* PoP_k after running Π_{PoP} according to lemma 4. Consequently, no honest party adds P_k to its list H_i after running $\Pi_{\text{d-MCoD}}$. Now, suppose two honest parties P_i and P_j to output different values $y_i \neq y_j$ along with respective grades $g_i = g_j = 0$ and faulty lists $\mathcal{F}^\star_i$ and $\mathcal{F}^\star_j$. P_i determines y_i as the majority bit over values $y_{i,j}$ output from $\Pi^j_{\text{d-MCoD}}$ where $P_j \in H_i$. The majority over these values can only differ in the view of P_j if H_i and H_j differ (e.g., a party P_k is in H_i but not in H_j) or if an instance of $\Pi^k_{\text{d-MCoD}}$ outputs different values $y_{i,k} \neq y_{j,k}$ for P_i and P_j. In the first case, the PoP PoP_i is valid for P_i and invalid for P_j. Thus, $\Pi_{\text{d-MCoD}}$ must have output different values $\mathsf{PoP}_i \neq \mathsf{PoP}_j$. In both cases, the d-detection property of $\Pi_{\text{d-MCoD}}$ ensures that at least d malicious parties are added to the faulty list of every honest party when they take the union of the faulty lists output in all instances of $\Pi^k_{\text{d-MCoD}}$. Additionally, these d malicious parties were not initially included in all honest parties' faulty lists, due to the reasons stated at the beginning of this proof.

Finally, we prove soundness and termination.

Lemma 8. *$\Pi_{\text{1-GDA}}$ achieves soundness, and termination as per Definition 4.*

Proof. Assume that for each honest party P_i, $P_i \notin \mathcal{F}_j$ for any honest party P_j.
Soundness. An honest party P_i adds additional parties to its initial faulty list $\mathcal{F}_i$ by including the parties from the union of all the faulty lists generated by the $\Pi^j_{\text{d-MCoD}}$ instances for each $P_j \in \mathcal{P}$. According to the $\mathcal{F}$-soundness of $\Pi_{\text{d-MCoD}}$, the resulting $\mathcal{F}^\star_i$ will only include malicious parties.
Termination. $\Pi_{\text{1-GDA}}$ is constructed from concurrent instances $\Pi_{\text{d-MCoD}}$. Based

on the assumption that $\Pi_{\text{d-MCoD}}$ terminates, $\Pi_{\text{1-GDA}}$ will also terminate.
Round Complexity. $\Pi_{\text{1-GDA}}$ protocol runs for $d+5$ rounds: one round for Π_{PoP} and $d+4$ for $\Pi_{\text{d-MCoD}}$. □

We summarize the previous lemmata into the main following Lemma of this section:

Lemma 9. *$\Pi_{\text{1-GDA}}$, (Fig. 4) achieves $(0,1)$-Graded d-Detecting Agreement as per Definition 5. Furthermore, $\Pi_{\text{1-GDA}}$ terminates in $d+5$ rounds.*

3.4 Deterministic Early-Stopping Byzantine Agreement Protocol (Π_{BA^d})

In this subsection, we demonstrate how to construct the deterministic early-stopping Byzantine agreement protocol, Π_{BA^d}, using $\Pi_{\text{1-GDA}}$. In Π_{BA^d}, each party starts with an input value $v_i \in \{0,1\}$ and outputs an output value $y_i \in \{0,1\}$. Π_{BA^d} runs in iterations. In each iteration k, parties run $\Pi_{\text{1-GDA}}$ with input $(v_i, \mathcal{F}_i)$. Consequently, each party P_i stores the output $(y_i, g_i, \mathcal{F}_i)$ of $\Pi_{\text{1-GDA}}$.

Based on the grade g_i obtained from $\Pi_{\text{1-GDA}}$, each party P_i determines whether it is safe to terminate. If P_i outputs $g_i = 0$, it indicates that it is not safe to terminate, and more iterations are required. P_i updates its input value for the next iteration based on the output value $y_i \in \{0,1\}$ of $\Pi_{\text{1-GDA}}$, setting $v_i = y_i$. Conversely, if P_i outputs $g_i = 1$, it is confident that all other honest parties P_j output the same value $y_i = y_j$ due to the graded consistency of $\Pi_{\text{1-GDA}}$. In this case, P_i runs for one more iteration to ensure that other honest parties can also safely terminate on the same value, as proven in Lemma 13. Note, a party can set its output value y_i in iteration k, but terminates a few iterations later. A party only terminates when $\mathsf{halt}_i = \mathit{true}$.

Each iteration consists of $d+5$ rounds: $d+5$ rounds for $\Pi_{\text{1-GDA}}$. Therefore, the overall round complexity of Π_{BA^d} depends on the number of iterations it runs. We demonstrate in Lemma 14 that the number of iterations is a function of f.

First, we establish that honest parties are never included in the faulty lists of other honest parties in any iteration. From this point forward, we assume this lemma holds indefinitely. Consequently, the assumption of $\Pi_{\text{1-GDA}}$ as stated in Definition 4 is always valid, and we may omit it from proofs for simplicity.

Lemma 10. *At the start of each iteration k of Π_{BA^d}, the faulty list $\mathcal{F}_i$ of every honest party P_i contains only corrupted parties.*

Proof. In the first iteration $k = 1$, the faulty lists of all honest parties are empty, so the lemma holds trivially. For subsequent iterations $k > 1$, each party updates its $\mathcal{F}_i$ based on the output of $\Pi_{\text{1-GDA}}$. According to the soundness property of $\Pi_{\text{1-GDA}}$, no honest party P_i is included in the $\mathcal{F}_j$ of any other honest party P_j in any of these iterations. Thus the claim follows by a simple induction. □

Next, we prove that if all honest parties set y_i to the same value in iteration k, all honest parties terminate by at most iteration $k+2$.

Protocol Π_{BA^d}

- **Input and Initialization:** Let v_i denote P_i's input. P_i sets $\mathsf{halt}_i := \mathsf{false}, y_i := \perp, \mathsf{wait}_i := \infty$ $\mathcal{F}_i := \emptyset$
- **While** $\mathsf{halt}_i = \mathsf{false}$ **do**
  - **Rounds** 1 **to** $d + 5$**:**
    - P_i runs protocol $\Pi_{\text{1-GDA}}$ with input $(v_i, \mathcal{F}_i)$ and stores output $(y_i, g_i, \mathcal{F}_i) := \Pi_{\text{1-GDA}}$
    - If $g_i = 1$ and $\mathsf{wait}_i = 1$, P_i sets $\mathsf{halt}_i := \mathit{true}$. Otherwise, if $g_i = 1$ and $\mathsf{wait}_i > 1$, P_i sets $\mathsf{wait}_i = 1$
    - Each party P_i updates the input of next iteration by setting $v_i := y_i$
- **Output Determination:** If $\mathsf{halt}_i = \mathit{true}$, P_i outputs y_i and terminates.

Fig. 5. Code of Π_{BA^d} for party P_i.

Lemma 11. *If all honest parties set y_i to the same value in iteration k, then all honest parties will terminate by at most iteration $k + 2$.*

Proof. Let all honest parties P_i set y_i to the same value v in iteration k. Each party then updates its input value v_i for the subsequent iteration based on this output value, such that $v_i = y_i$. In the next iteration $(k + 1)$, all honest parties invoke $\Pi_{\text{1-GDA}}$ with this updated input value v_i. According to the validity of $\Pi_{\text{1-GDA}}$, all honest parties will set $g_i = 1$ in iteration $k + 1$ and will terminate in iteration $k + 2$. □

Next, we proceed with proving validity and consistency for Π_{BA^d}.

Lemma 12. *Π_{BA^d} achieves validity per Definition 1*

Proof. Assume all honest parties have the same initial value ($v_i = \mathsf{v}$). Every party invokes $\Pi_{\text{1-GDA}}$ with input $(v_i, \mathcal{F}_i)$. From graded validity of $\Pi_{\text{1-GDA}}$, every honest party outputs $y_i = \mathsf{v}$ and $g_i = 1$. Consequently, every honest party terminates by the end of iteration $k+1$ with $y_i = \mathsf{v}$. In the subsequent iteration $(k+1)$, each party invokes $\Pi_{\text{1-GDA}}$ with $v_i = \mathsf{v}$. Thus, no party updates its output variable to $\mathsf{v}' \neq \mathsf{v}$ due to graded validity of $\Pi_{\text{1-GDA}}$. □

Lemma 13. *Π_{BA^d} achieves consistency per Definition 1*

Proof. Let P_i denote the first honest party that sets $\mathsf{wait}_i = 1$ in the earliest iteration, say $k > 0$, indicating it will *wait* for one more iteration before terminating. This occurs when p_i sets its g_i to 1, determined by the output of $\Pi_{\text{1-GDA}}$ in iteration k. According to the graded consistency of $\Pi_{\text{1-GDA}}$, every other honest party P_j outputs $y_j = \mathsf{v}$. Consequently, every honest party updates its input variable for the next iteration to $v_i = \mathsf{v}$. Therefore, in iteration $k + 1$, all honest parties have the same input value for $\Pi_{\text{1-GDA}}$. Due to the graded validity, all honest parties set $y_i = \mathsf{v}$ and $g_i = 1$, and they all terminate by iteration $k + 2$ at the latest. □

Finally, we prove that Π_{BA^d} terminates in $(d+5)\cdot(\lceil f/d\rceil+2)$ rounds.

Lemma 14. *Π_{BA^d} terminates in $(d+5)\cdot(\lfloor f/d\rfloor+3)$ rounds.*

Proof. In any iteration k, if honest parties P_i and P_j have the same output value $y_i = y_j$ based on the output of $\Pi_{\text{1-GDA}}$, then all honest parties will terminate by iteration $k+2$, as proven in Lemma 11. If in some iteration, P_i and P_j have different output values, i.e., $y_i \neq y_j$ from $\Pi_{\text{1-GDA}}$, then according to the d-detection property of $\Pi_{\text{1-GDA}}$, at least d malicious parties are added to the faulty list $\mathcal{F}_i$ of all honest parties $P_i \in \mathcal{P}$. Thus, since there are f faulty parties, there can be at most $\lfloor f/d\rfloor$ many iterations where there are distinct honest parties P_i and P_j that output different values $y_i \neq y_j$ from $\Pi_{\text{1-GDA}}$. Thus, after at most $\lfloor f/d\rfloor+1$ many iterations, all honest parties output the same value $\mathsf{v} \in \{0,1\}$. Hence, they all terminate by iteration $\lfloor f/d\rfloor+3$ by Lemma 11. Since each iteration takes $d+5$ rounds, the overall complexity comes out to $(d+5)\cdot(\lfloor f/d\rfloor+3))$. □

We summarize Lemma 12, 13, and 14 as the following Theorem 1:

Theorem 1. *Assume a PKI setup and $t < n/2$. Π_{BA^d} (Fig. 5) achieves Byzantine Agreement per Definition. 1. Furthermore, Π_{BA^d} terminates in $(d+5)\cdot(\lfloor f/d\rfloor+3)$ rounds, for any execution with $f \leq t$ corrupted parties and runs in communication complexity $O(f\cdot n^4)$.*

Proof. The theorem follows from the preceeding lemmata. For the communication complexity, we note that the complexity of an instancce of $\Pi_{\text{d-CoD}}$ is $O(n^2\cdot d)$ and during each iteration of Π_{BA^d}, $O(n^2)$ such instances are called to broadcast the PoPs of length $O(n)$ bit by bit for $O(n)$ senders. Since the protocol has $O(f/d)$ iterations, the overall complexity is $O(n^2\cdot n^2\cdot d\cdot f/d) = O(n^4\cdot f)$.

4 Byzantine Agreement with Expected Constant and Worst-Case Early-Stopping Round Complexity

In this section, we introduce our randomized Byzantine Agreement protocol, Π_{BA^r}, which achieves both expected constant time and worst-case early-stopping round complexity. Similar to our deterministic protocol, Π_{BA^r} is built using the $(0,1,2)$-Graded d-Detecting Agreement protocol, $\Pi_{\text{2-GDA}}$. Therefore, we begin by introducing $\Pi_{\text{2-GDA}}$ and then present the complete construction of Π_{BA^r}.

4.1 $(0,1,2)$-Graded d-Detecting Agreement ($\Pi_{\text{2-GDA}}$)

Similar to $\Pi_{\text{1-GDA}}$ protocol, $\Pi_{\text{2-GDA}}$ is a variant of Graded Consensus protocols[11], which allows honest parties to also output a list of detected malicious parties. In $\Pi_{\text{2-GDA}}$, each party starts with $v_i \in \{0,1\}$ and faulty list $\mathcal{F}_i$. Each party outputs a value $y_i \in \{0,1,\perp\}$, a grade $g_i \in \{0,1\}$, and an updated list of identified malicious parties $\mathcal{F}_i^\star \subset \mathcal{P}$. $\Pi_{\text{2-GDA}}$ is constructed from $\Pi_{\text{1-GDA}}$ and the black-box

$(0,1,2)$-Graded Agreement protocol from [23], $\Pi_{\text{2-GA}}$, which we include in the Appendix. In the first round, each party P_i invokes $\Pi_{\text{1-GDA}}$ with input $(v_i, \mathcal{F}_i)$, storing the resulting output $(y_i^\star, g_i^\star, \mathcal{F}_i^\star)$. To enhance the confidence on its output value, the parties run $\Pi_{\text{2-GA}}$ with $y_i^\star$ as its input. Finally, party P_i terminates and outputs $(y_i, g_i, \mathcal{F}_i^\star)$, where they are the output of $\Pi_{\text{2-GA}}$. Note that the output $\mathcal{F}_i^\star$ is the faulty list output from $\Pi_{\text{1-GDA}}$ and does not get updated further. The protocol runs for $d+9$ rounds: $d+5$ for $\Pi_{\text{1-GDA}}$ and 4 additional rounds for $\Pi_{\text{2-GA}}$.

Protocol $\Pi_{\text{2-GDA}}$

- **Input and Initialization:** Let v_i and $\mathcal{F}_i$ denote P_i's input. P_i sets $y_i, y_i^\star := \perp, g_i, g_i^\star := 0, \mathcal{F}_i^\star := \emptyset$
- **Rounds $r = 1$ to $d+5$:**
  - P_i invokes $\Pi_{\text{1-GDA}}$ with input $(v_i, \mathcal{F}_i)$. Let $(y_i^\star, g_i^\star, \mathcal{F}_i^\star)$ denote the output.
- **Rounds $r = d+6$ to $r = d+9$:**
  - P_i invokes $\Pi_{\text{2-GA}}$ with input $y_i^\star$ and let (y_i, g_i) denote the output.
- **Output Determination:** P_i outputs $(y_i, g_i, \mathcal{F}_i^\star)$ and terminates

Fig. 6. Code of $\Pi_{\text{2-GDA}}$ for party P_i.

Lemma 15. *Assume $\Pi_{\text{2-GA}}$ achieves $(0,1,2)$-Graded Agreement per Definition 2. $\Pi_{\text{2-GDA}}$ achieves $(0,1,2)$-Graded Faulty-Detecting Byzantine Agreement per Definition 5.*

Proof. Assume that for each honest party P_i, $P_i \notin \mathcal{F}_j$ for any honest party P_j. Suppose that every honest party P_i inputs $(v_i, \mathcal{F}_i)$ to $\Pi_{\text{2-GDA}}$, where $v_i \in \{0,1\}$ and $\mathcal{F}_i \subset \mathcal{P}$.

Graded Validity. By assumption, every honest party starts with $v_i = \mathsf{v}$, and invokes $\Pi_{\text{1-GDA}}$ with input $(\mathsf{v}, \mathcal{F}_i)$. According to the graded validity of $\Pi_{\text{1-GDA}}$ (Definition 4), all honest parties outputs $(\mathsf{v}, 1, \mathcal{F}_i^\star)$. Thus in round $d+6$, every honest party invokes $\Pi_{\text{2-GA}}$ with input v. From graded validity of $\Pi_{\text{2-GA}}$, P_i outputs $y_i = \mathsf{v}$ and $g_i = 2$.

Graded Consistency. A party p_i sets its g_i and y_i based on the output of $\Pi_{\text{2-GA}}$. From graded consistency of $\Pi_{\text{2-GA}}$, this holds.

d-Detection. Assume an honest party p_i outputs a $g_i < 2$. If an honest party p_i outputs a $g_i < 2$, it follows from graded validity of $\Pi_{\text{2-GA}}$ that not all parties input the same value to $\Pi_{\text{2-GA}}$. Parties invoke $\Pi_{\text{2-GA}}$ with the output value they obtained from $\Pi_{\text{1-GDA}}$, so there must be two honest parties P_i and P_j that output distinct values y_i^* and y_j^* from $\Pi_{\text{1-GDA}}$. Thus, d-detection of $\Pi_{\text{2-GDA}}$ is directly implied by d-detection of $\Pi_{\text{1-GDA}}$.

Soundness. The output faulty list, denoted as $\mathcal{F}_i^\star$, is based on the output faulty list from $\Pi_{\text{1-GDA}}$. Due to the soundness property of $\Pi_{\text{1-GDA}}$, $\mathcal{F}_i^\star$ contains only malicious parties.

Termination: The protocol invokes $\Pi_{\text{1-GDA}}$ and $\Pi_{\text{2-GA}}$, which terminates as per definitions 4 and 2 respectively.

4.2 Byzantine Agreement with Expected Constant and Worst-Case Early-Stopping Round Complexity

In this subsection, we present our randomized Byzantine agreement protocol which has expected constant time and worst case early-stopping round complexity. We demonstrate how to construct the randomized early-stopping Byzantine agreement protocol, Π_{BA^r}, using $\Pi_{\text{2-GDA}}$. In Π_{BA^r}, each party starts with an input value $v_i \in \{0,1\}$ and outputs an output value $y_i \in \{0,1\}$. Π_{BA^r} runs in iterations. In each iteration k, parties run $\Pi_{\text{2-GDA}}$ with input $(v_i, \mathcal{F}_i)$. Consequently, each party P_i stores the output $(y_i, g_i, \mathcal{F}_i)$ of $\Pi_{\text{2-GDA}}$. Based on the grade g_i obtained from $\Pi_{\text{2-GDA}}$, each party P_i determines whether it is safe to terminate. If P_i outputs $g_i < 2$, it indicates that it is not safe to terminate, and more iterations are required.

Conversely, if P_i outputs $g_i = 2$, it is confident that all other honest parties P_j output the same value $y_i = y_j$ due to the graded consistency of $\Pi_{\text{2-GDA}}$. Party P_i then updates its input value for the next iteration based on the output grade $g_i \in \{0,1,2\}$ of $\Pi_{\text{2-GDA}}$. If $g_i > 0$, it updates its input value to the next iteration based on the output value $y_i \in \{0,1\}$ of $\Pi_{\text{2-GDA}}$, setting $v_i = y_i$. Otherwise, if $g_i = 0$, it sets its input value to the next iteration based on the random coin it receives from the $\mathsf{CoinFlip}$ protocol. We show in lemma 21 that Π_{BA^r} has expected constant time.

Each iteration consists of $d+9$ rounds due to $\Pi_{\text{2-GDA}}$ protocol. Therefore, the overall round complexity of Π_{BA^r} depends on the number of iterations it runs in the worst case. We demonstrate in Lemma 14 that the number of iterations in the worst case is a function of f.

Similar to $\Pi_{\text{1-GDA}}$, we also establish that honest parties are never included in the faulty lists of other honest parties in any iteration, which is a needed assumption for $\Pi_{\text{2-GDA}}$

Lemma 16. *At the start of each iteration, the faulty list $\mathcal{F}_i$ of every honest party P_i contains only corrupted parties.*

Proof. The proof is derived from Lemma 10 and the fact that $\mathcal{F}_i$ in Π_{BA^r} is based on the faulty list produced by $\Pi_{\text{2-GDA}}$. □

We proceed to prove both validity of Π_{BA^r}.

Lemma 17. *Π_{BA^r} achieves validity per Definition 1*

Proof. Assume all honest parties have the same initial value ($v_i = \mathsf{v}$). Every party invokes $\Pi_{\text{2-GDA}}$ in the second round with input $(v_i, \mathcal{F}_i)$. From graded validity of $\Pi_{\text{2-GDA}}$, every honest party outputs $(y_i = \mathsf{v}, g_i = 2, \mathcal{F}_i)$. Consequently, every honest party terminates by the end of iteration $k+1$ with $y_i = \mathsf{v}$. In the subsequent iteration $(k+1)$, each party invokes $\Pi_{\text{2-GDA}}$ with $v_i = \mathsf{v}$ Thus, no party updates its output variable to $\mathsf{v}' \neq \mathsf{v}$ due to graded validity. □

Protocol Π_{BA^r}

- **Input and Initialization:** Let v_i denote P_i's input. P_i sets $k := 0$ $\mathsf{halt}_i := \mathsf{false}, y_i := \perp, \mathsf{wait}_i := \infty$ $\mathcal{F}_i := \emptyset$
- **While** $\mathsf{halt}_i = \mathsf{false}$ **do**
  - $k := k + 1$
  - **Rounds** 1 **to** $d + 9$**:**
    - P_i invokes protocol $\Pi_{\text{2-GDA}}$ with input $(v_i, \mathcal{F}_i)$. Let $(y_i, g_i, \mathcal{F}_i)$ denote the output.
    - P_i updates the input for next iteration $v_i := y_i$
    - If $g_i = 2$ and $\mathsf{wait}_i = 1$, P_i sets $\mathsf{halt}_i := \mathit{true}$. Otherwise, if $g_i = 2$ and $\mathsf{wait}_i > 1$, P_i sets $\mathsf{wait}_i = 1$
    - If $g_i = 0$, party P_i updates the next iteration's input using the common coin, $c_i^{(k)} \leftarrow \mathsf{CoinFlip}(k)$. It sets $v_i := c_i^{(k)}$.
- **Output Determination:** If $\mathsf{halt}_i = \mathit{true}$, P_i outputs y_i and terminates.

Fig. 7. Code of Π_{BA^r} for party P_i

Next, we prove consistency.

Lemma 18. *Π_{BA^d} achieves consistency per Definition 1*

Proof. Let p_i be the first honest party to set $\mathsf{wait}_i = 1$ in the earliest iteration, say $k > 0$, indicating it will wait for one more iteration before terminating. This happens when p_i sets wait_i to 1, a condition met if its g_i equals 2, determined by the output of $\Pi_{\text{2-GDA}}$. By the graded consistency of $\Pi_{\text{2-GDA}}$, every other honest party p_j outputs $y_j = \mathsf{v}$ and $g_j \geq 1$. As a result, every honest party updates its input variable for the next iteration to $v_i = \mathsf{v}$. Therefore, in iteration $k + 1$, all honest parties invoke $\Pi_{\text{2-GDA}}$ with input $(v_i = \mathsf{v})$. Due to the graded validity, all honest parties set $y_i = \mathsf{v}$ and $g_i = 2$, and they all terminate in the subsequent iteration.

Next, we show two lemmata that will help us in ultimately proving the round complexity.

Lemma 19. *If an honest party sets its g_i to 2 in some iteration k, then all honest parties will terminate by iteration $k + 3$ at the latest.*

Proof. Let honest party P_i set its grade $g_i = 2$ in iteration k and $y_i = \mathsf{v}$. From graded consistency of $\Pi_{\text{2-GDA}}$, every honest party P_j sets $y_j = \mathsf{v}$. Consequently, every honest party updates its input variable to the next iteration $v_i = \mathsf{v}$. Consequently, in iteration $k+1$, all honest parties invoke $\Pi_{\text{2-GDA}}$ with input $(v_i = \mathsf{v})$. Due to the graded validity, all honest parties set $y_i = \mathsf{v}$ and $g_i = 2$. And, they all terminate in the subsequent iteration.

Lemma 20. *If all honest parties set y_i to the same value in iteration k, then all honest parties will terminate by at most iteration $k + 2$.*

Proof. The proof follows similar logic to Lemma 11. □

Finally, we prove that Π_{BA^r} terminates in expected constant time and $(d+9)\cdot(\lfloor f/d\rfloor+2)$ rounds in the worst case.

Lemma 21. *Π_{BA^d} has expected constant time and always terminating within $(d+9)\cdot(\lfloor f/d\rfloor+2)$ rounds.*

Proof. The proof follows a similar approach to that used in theorem 1. First, we demonstrate the worst-case round complexity. A party terminates one iteration after setting its g_i to 2, and all other honest parties terminate after two more iterations from Lemma 19. The setting of g_i by a party is based on the result of $\Pi_{\text{2-GDA}}$. If an honest party P_i sets $g_i < 2$, then at least d parties are added to the faulty list of all honest parties P_i according to the d-detection property of $\Pi_{\text{2-GDA}}$. Thus, since there are f faulty parties, there can be at most $\lfloor f/d\rfloor$ many iterations where all honest parties output $g_i < 2$. Thus, after at most $\lfloor f/d\rfloor + 1$ many iterations, all honest parties set $g_i = 2$, followed by one additional iteration for all honest parties to terminate. Therefore, the total worst-case round complexity is $(d+9)\cdot(\lfloor f/d\rfloor+2)$. Next, we prove expected constant time. If an honest party P_i has $g_i = 2$ by the end of iteration k, all honest parties terminate by the end of iteration $k+2$. So, let's assume every honest party has $g_i < 2$ by iteration k. Then, with a probability of at least $1/2\cdot p$, the common coin value $c_j^{(k)}$ of all honest parties $P_j \in \mathcal{P}$ is equal to the y_i of honest parties P_i with $g_i = 1$. Thus, all honest parties start the next iteration with the same value. From Lemma 20, all honest parties terminate by iteration $k+2$. Thus, the exact round complexity in expectation is $((2/p)+2)(d+9)$

We summarize the preceding lemmata into the main theorem of this section:

Theorem 2. *Assume a PKI setup, random common coin, and $t < n/2$. Π_{BA^r} (Fig. 7) achieves Byzantine Agreement per Definition. 1. Furthermore, Π_{BA^r} terminates in expected constant time and worst case $(d+9)\cdot(\lfloor f/d\rfloor+2)$ rounds, for any execution with $f \le t$ corrupted parties and runs in communication complexity $O(f\cdot n^4)$.*

Proof. The theorem follows from the preceding lemmata. For the communication complexity, we established that the deterministic protocol runs in communication complexity $O(n^4\cdot d)$. The protocol Π_{BA^r} runs four additional rounds per iteration compared to Π_{BA^d} due to the construction of $\Pi_{\text{2-GDA}}$. These extra rounds run $\Pi_{\text{2-GA}}$, which has a communication complexity of $O(n^3)$ [23]. Thus, the overall complexity of Π_{BA^r} stays $O(n^4\cdot f)$.

References

1. K. Perry and S. Toueg, "An authenticated byzantine generals algorithm with early stopping," *Cornell eCommons*, 1984, online.

2. O. Goldreich and E. Petrank, "The best of both worlds: guaranteeing termination in fast randomized byzantine agreement protocols," *Information Processing Letters*, vol. 36, no. 1, pp. 45–49, 1990. [Online]. Available: https://www.sciencedirect.com/science/article/pii/002001909090185Z
3. L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, p. 382–401, jul 1982. [Online]. Available: https://doi.org/10.1145/357172.357176
4. D. Dolev and H. Strong, "Authenticated algorithms for byzantine agreement," *SIAM J. Comput.*, vol. 12, pp. 656–666, 11 1983.
5. D. Dolev, R. Reischuk, and H. R. Strong, "Early stopping in byzantine agreement," *J. ACM*, vol. 37, no. 4, p. 720–741, oct 1990. [Online]. Available: https://doi.org/10.1145/96559.96565
6. I. Abraham and D. Dolev, "Byzantine agreement with optimal early stopping, optimal resilience and polynomial complexity," in *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 605–614. [Online]. Available: https://doi.org/10.1145/2746539.2746581
7. M. Fitzi and J. Nielsen, "On the number of synchronous rounds sufficient for authenticated byzantine agreement," 09 2009, pp. 449–463.
8. P. Berman, J. A. Garay, and K. J. Perry, "Bit optimal distributed consensus," *Computer Science*, pp. 313–321, 1992.
9. J. A. Garay and Y. Moses, "Fully polynomial byzantine agreement in t + 1 rounds," in *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '93. New York, NY, USA: Association for Computing Machinery, 1993, p. 31–41. [Online]. Available: https://doi.org/10.1145/167088.167101
10. Garay and Y. Moses, "Fully polynomial byzantine agreement for n > 3t processors in t + 1 rounds," *SIAM Journal on Computing*, vol. 27, no. 1, pp. 247–290, 1998. [Online]. Available: https://doi.org/10.1137/S0097539794265232
11. P. Feldman and S. Micali, "Optimal algorithms for byzantine agreement," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. New York, NY, USA: Association for Computing Machinery, 1988, p. 148–161. [Online]. Available: https://doi.org/10.1145/62212.62225
12. I. Abraham, T.-H. H. Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi, "Communication complexity of byzantine agreement, revisited," 2020.
13. J. Wan, H. Xiao, S. Devadas, and E. Shi, "Round-efficient byzantine broadcast under strongly adaptive and majority corruptions," in *Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part I.* Berlin, Heidelberg: Springer-Verlag, 2020, p. 412–456. [Online]. Available: https://doi.org/10.1007/978-3-030-64375-1_15
14. J. Katz and C.-Y. Koo, "On expected constant-round protocols for byzantine agreement," in *Advances in Cryptology - CRYPTO 2006*, C. Dwork, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 445–462.
15. A. Zamsky, "An randomized byzantine agreement protocol with constant expected time and guaranteed termination in optimal (deterministic) time," in *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 201–208. [Online]. Available: https://doi.org/10.1145/248052.248091
16. P. R. Parvédy and M. Raynal, "Optimal early stopping uniform consensus in synchronous systems with process omission failures," in *Proceedings of the*

Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, ser. SPAA '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 302–310. [Online]. Available: https://doi.org/10.1145/1007912.1007963
17. K. Alpturer, J. Y. Halpern, and R. van der Meyden, "Optimal eventual byzantine agreement protocols with omission failures," in *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, ser. PODC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 244–252. [Online]. Available: https://doi.org/10.1145/3583668.3594573
18. A. Castañeda, Y. Moses, M. Raynal, and M. Roy, "Early decision and stopping in synchronous consensus: a predicate-based guided tour," in *International Conference on Networked Systems*. Springer, 2017, pp. 206–221.
19. J. Loss and J. B. Nielsen, "Early stopping for any number of corruptions," in *Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part III*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 457–488. [Online]. Available: https://doi.org/10.1007/978-3-031-58734-4_16
20. R. Canetti, "Universally composable security," *J. ACM*, vol. 67, no. 5, sep 2020. [Online]. Available: https://doi.org/10.1145/3402457
21. S. Micali, "Very simple and efficient byzantine agreement," in *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, ser. LIPIcs, C. H. Papadimitriou, Ed., vol. 67. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 6:1–6:1. [Online]. Available: https://doi.org/10.4230/LIPIcs.ITCS.2017.6
22. C. Cachin, K. Kursawe, and V. Shoup, "Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography," Cryptology ePrint Archive, Paper 2000/034, 2000, https://eprint.iacr.org/2000/034. [Online]. Available: https://eprint.iacr.org/2000/034
23. M. Fitzi and U. Maurer, "From partial consistency to global broadcast," in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, ser. STOC '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 494–503. [Online]. Available: https://doi.org/10.1145/335305.335363

A Supplementary Material

A.1 (0, 1, 2)-Graded Broadcast

We present the $(0, 1, 2)$-graded agreement protocol [23] that we use as a subroutine in the $(0, 1, 2)$-Graded d-Detecting Agreement. We then construct graded agreement from graded broadcast of [14], $\Pi_{\text{2-GB}}$ (Fig 9). We first show $\Pi_{\text{2-GB}}$ in Fig. 8 and refer the reader to [14] for the full correctness proof.

Protocol $\Pi_{\text{2-GB}}$

- **Input and Initialization:** If $P_i = P_s$, let v_s denote P_s's input. P_i sets $y_i := \perp, g_i := 0, m_i = \perp$
- **Round 1:**
  - If $P_i = P_s$, it sends $\langle v_s \rangle_s$ to all parties.
- **Round 2:**
  - If P_i received $\langle v_s \rangle_s$ in the previous round, it sets $m_i := \langle v_s \rangle_S$, and forwards m_i to all parties. Otherwise, does nothing.
- **Round 3:**
  - Let $m_{i,j}$ be the message received by P_i from P_j in the previous round. If $\exists m_{i,j}$ such that $m_{i,j} \neq m_i$, P_i sets $m_i := \perp$. Otherwise, it sends m_i to all parties
- **Round 4:**
  - Let $m'_{j.i}$ be the message received by P_i from P_j in the previous round. If $\exists$ at least distinct $l > n/2$ received messages $m'_{j,i}$ for $j \in [n]$, where $m'_{j_1,i} = \cdots = m'_{j_l,i} = \langle \mathsf{v} \rangle_s$, P_i sets $y_i := \mathsf{v}$ and $g_i = 2$. Furthermore, P_i sends the l messages to all parties.
- **Output Determination:** Assume P_i has not set its output; $y_i = \perp$, it proceeds as follows. If in the previous round P_i receives $l > n/2$ distinct messages $m'_{j,i}$ for $j \in [n]$, where $m'_{j_1,i} = \cdots = m'_{j_l,i} = \langle \mathsf{v} \rangle_s$, P_i sets $y_i := \mathsf{v}$ and $g_i := 1$. Otherwise P_i sets $g_i := 0$ and $y_i := \perp$.

Fig. 8. Code of $\Pi_{\text{2-GB}}$ for party P_i.

A.2 (0, 1, 2)-Graded Agreement

Next, to achieve graded agreement from graded broadcast, each party invokes a graded broadcast with its input v_i. As a result, each party determines the overall grade and output value based on the output values and grades from all the invoked graded broadcast protocols. The construction is shown in Fig. 9

Protocol $\Pi_{\text{2-GA}}$

- **Input and Initialization:** Let v_i denote P_i's input. P_i sets $y_i := \perp$, $g_i := 0$, and $y_{i,j} = \perp, g_{i,j} := 0$ for $j \in [n]$
- **Round $r = 1$ to 4:**
  - P_i invokes $\Pi_{\text{2-GB}}$ with input v_i.
- **Output Determination:** Let $(y_{i,j}, g_{i,j})$ denote the output for party P_i of $\Pi^j_{\text{2-GB}}$ with party P_j as sender. If at least $t + 1$ instances output $y_{i,j} = \mathsf{v}$ and $g_{i,j} = 2$, P_i sets $y_i = \mathsf{v}$ and $g_i = 2$. Else, if at least $t + 1$ instances output $y_{i,j} = \mathsf{v}$ and $g_{i,j} \in \{1, 2\}$, it sets $y_i = \mathsf{v}$ and $g_i = 1$. Otherwise, it outputs $g_i = 0$ and $y_i = \perp$

Fig. 9. Code of $\Pi_{\text{2-GA}}$ for party P_i.